LAB MANUAL OF
OBJECT ORIENTED PROGRAMMING
ETCS 258

Maharaja Agrasen Institute of Technology, PSP area,
Sector – 22, Rohini, New Delhi – 110085

(Affiliated to Guru Gobind Singh Indraprastha University,
New Delhi)

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

VISION

To nurture young minds in a learning environment of high academic value and imbibe spiritual and ethical values with technological and management competence.

MISSION

The Institute shall endeavor to incorporate the following basic missions in the teaching methodology:

Engineering Hardware – Software Symbiosis

Practical exercises in all Engineering and Management disciplines shall be carried out by Hardware equipment as well as the related software enabling deeper understanding of basic concepts and encouraging inquisitive nature.

Life – Long Learning

The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning.

Liberalization and Globalization

The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.

Diversification

The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.

Digitization of Learning Processes

The Institute provides seamless opportunities for innovative learning in all Engineering and Management disciplines through digitization of learning processes using analysis, synthesis, simulation, graphics, tutorials and related tools to create a platform for multi-disciplinary approach.

Entrepreneurship

The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they become successful entrepreneurs and responsible citizens.

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

VISION

To Produce "Critical thinkers of Innovative Technology"

MISSION

To provide an excellent learning environment across the computer science discipline to inculcate professional behavior, strong ethical values, innovative research capabilitiesand leadership abilities which enable them to become successful entrepreneurs in this globalized world.

1. To nurture an **excellent learning environment** that helps students to enhance their problem solving skills and to prepare students to be lifelong learners by offering a solid theoretical foundation with applied computing experiences and educating them about their **professional, and ethical responsibilities**.
2. To establish **Industry-Institute Interaction**, making students ready for the industrial environment and be successful in their professional lives.
3. To promote **research activities** in the emerging areas of technology convergence.
4. To build engineers who can look into technical aspects of an engineering solution thereby setting a ground for producing successful **entrepreneur.**

INDEX OF THE CONTENTS

1. Introduction to the Lab

Lab Objective

The purpose of Object Oriented Programming is to add object orientation to the C programming language, which is in itself one of the most powerful programming language.

The core of the object-oriented programming is to create an object, in code, that has certain properties and methods. While designing C++ modules, we try to see whole world in the form of objects.

Course Outcomes

At the end of the course, a student will be able to:

C258.1: Perform object oriented programming to develop solutions to problems demonstrating usage of control structures, modularity, I/O and other standard language constructs.

C258.2: Demonstrate adeptness of object oriented programming in developing solutions to problems demonstrating usage of data abstraction, encapsulation, and inheritance.

C258.3: Learn on the special member functions of the class and demonstrate on creation and deletion of class objects.

C258.4: Demonstrate ability to implement one or more patterns involving realization of an abstract interface and utilization of polymorphism in the solution of problems which can take advantage of dynamic binding.

C258. 5: Learn syntax and features of the utilization of Standard Template library.

C258. 6: Demonstrate Opening and Closing of a File and how to read and write in to a file.

2. LAB REQUIREMENTS

Hardware Detail

Intel i3/C2D Processor/2 GB RAM/500GB HDD/MB/Lan Card/ Key Board/ Mouse/CD Drive/15" Color Monitor/ UPS	24 Nos
LaserJet Printer	1 No

Software Detail

For C++ Programming:

VI Editor or any other Text Editor

Code Block

3. LIST OF EXPERIMENTS (As prescribed by G.G.S.I.P.U)

Paper Code: ETCS-258
Paper: Object Oriented Programming Lab

List of Experiment:

1. Write a program for multiplication of two matrices using OOP.
2. Write a program to perform addition of two complex numbers using constructor overloading. The first constructor which takes no argument is used to create objects which are not initialized, second which takes one argument is used to initialize real and imag parts to equal values and third which takes two argument is used to initialized real and imag to two different values.
3. Write a program to find the greatest of two given numbers in two different classes using friend function.
4. Implement a class string containing the following functions:
   - Overload + operator to carry out the concatenation of strings.
   - Overload = operator to carry out string copy.
   - Overload <= operator to carry out the comparison of strings.
   - Function to display the length of a string.
   - Function tolower( ) to convert upper case letters to lower case.
   - Function toupper( ) to convert lower case letters to upper case.
5. Create a class called LIST with two pure virtual function store() and retrieve().To store a value call store and to retrieve call retrieve function. Derive two classes stack and queue from it and override store and retrieve.
6. Write a program to define the function template for calculating the square of given numbers with different data types.
7. Write a program to demonstrate the use of special functions, constructor and destructor in the class template. The program is used to find the bigger of two entered numbers.
8. Write a program to perform the deletion of white spaces such as horizontal tab, vertical tab, space ,line feed ,new line and carriage return from a text file and store the contents of the file without the white spaces on another file.
9. Write a program to read the class object of student info such as name , age ,sex ,height and weight from the keyboard and to store them on a specified file using read() and write() functions. Again the same file is opened for reading and displaying the contents of the file on the screen.
10. Write a program to raise an exception if any attempt is made to refer to an element whose index is beyond the array size.

NOTE:- At least 8 Experiments out of the list must be done in the semester.

4. LIST OF EXPERIMENTS (Beyond the syllabus)

Students will be divided into a group of four/five and projects are allotted to those groups. This project is to be submitted at the end of the semester along with a project report by the individual student.

List of projects given to the students is summarized as below:

1. Employee management system
2. Banking system
3. Library management system
4. Telephone exchange
5. Brick game

Students can select project work of his/her own choice subject to the permission of concern faculty.

NOTE: The project is to be made in C++ preferbly.

INSTRUCTIONS FOR PROJECT REPORT

1. TITLE :
2. MEMBERS IN THE PROJECT GROUP :
3. PROJECT REPORT ATTACHED :
   a) YES b) NO
4. SOFT COPY SUBMITTED :
   a) YES b) NO

Signature of the Faculty Signature of the student

( ) ( )

Each project which student is performing in the lab should have the following details :

a) Problem Statement
b) Problem Analysis
c) Source Code
d) Output

Project report should be added at last page.

5. FORMAT OF THE LAB RECORD TO BE PREPARED BY THE STUDENTS

The front page of the lab record prepared by the students should have a cover page as displayed below.

NAME OF THE LAB

Paper Code

Font should be(Size 20", italics bold, Times New Roman)

Faculty name

Student name

Roll No.:

Semester:

Font should be (12", Times Roman)

Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

Font should be (18", Times Roman)

Index

Exp. no	Experiment Name	Date of performance	Date of checking	Marks	Signature

6. MARKING SCHEME FOR THE PRACTICAL EXAMS

There will be two practical exams in each semester.

i. Internal Practical Exam
ii. External Practical Exam

INTERNAL PRACTICAL EXAM

It is taken by the respective faculty of the batch.

MARKING SCHEME FOR THIS EXAM IS:

Total Marks: 40

Division of 10 marks per practical is as follows:

	Rubrics for : Laboratory (General)			
Sr No.	Experiment Component (LAC)	Max. Marks	Grading Rubrics	
			2 marks	1 mark
1	Practical Performance	2	Completeness of practical, exhibits proficiency in using different types of inputs.	Incomplete practical, unformatted, lacks comments, Demonstrates no proficiency.
2	Output and Validation	2	Output is free of errors and output is obtained. Demonstrates excellent understanding of the concepts relevant to the experiment.	Output contains few logical errors and/or no output is obtained. Demonstrates partial understanding of the concepts relevant to the experiment.
3	Attendance and Viva Questions Answered	4	1. Four marks for answering more than 75% questions. 2. Two marks for answering more then 50% questions. 3. One mark for answering less then 50% questions.	
4	Timely Submission of Lab Record	2	On time submission	Late submission

Each experiment will be evaluated out of 10 marks. At the end of the semester average of 8 best performed practical will be considered as marks out of 40.

EXTERNAL PRACTICAL EXAM

It is taken by the concerned lecturer of the batch and by an external examiner. In this exam student needs to perform the experiment allotted at the time of the examination, a sheet will be given to the student in which some details asked by the examiner needs to be written and at the last viva will be taken by the external examiner.

MARKING SCHEME FOR THIS EXAM IS:

Total Marks: 60

Division of 60 marks is as follows

1. Sheet filled by the student: 20
2. Viva Voice: 15
3. Experiment performance: 15
4. File submitted: 10

NOTE:

- Internal marks + External marks = Total marks given to the students
  (40 marks) (60 marks) (100 marks)
- Experiments given to perform can be from any section of the lab.

7. INSTRUCTIONS FOR EACH LAB EXPERIMENT

Experiment 1

Aim: Write a program for multiplication of two matrices using OOP.

Performance Instructions:

1. The program takes two matrices and multiplies them.
2. If number of columns of matrix A is not equal to number of rows of matrix B, then matrices cannot be added.
3. The program is exited.
4. Else they are multiplied and the result is printed.
5. Exit.

```
// Enter rows and columns for matrix

cout << "Enter rows and columns for first matrix: ";
cin >> r1 >> c1;
cout << "Enter rows and columns for second matrix: ";
cin >> r2 >> c2;


// Storing elements of  matrix.

cout << endl << "Enter elements of matrix 1:" << endl;
for(i = 0; i < r1; ++i)
    for(j = 0; j < c1; ++j)
    {
        cout << "Enter element a" << i + 1 << j + 1 << " : ";
        cin >> a[i][j];
    }


// Multiplying matrix a and b and storing in array mult.

for(i = 0; i < r1; ++i)
    for(j = 0; j < c2; ++j)
        for(k = 0; k < c1; ++k)
        {
            mult[i][j] += a[i][k] * b[k][j];
        }
```

```
   // Displaying the multiplication of two matrix.

   cout << endl << "Output Matrix: " << endl;
   for(i = 0; i < r1; ++i)
   for(j = 0; j < c2; ++j)
   {
      cout << " " << mult[i][j];
      if(j == c2-1)
         cout << endl;
   }

   return 0;
}
```

Additional Programs:

1. Write a function power to raise a number m to power n. The function takes a double value for m and int value for n. Use default value for n to make the function to calculate squares when this argument is omitted.

2. Create a class TIME with members hours, minutes,seconds. Take input, add two time objects and pass objects as argument to function and display result.

Sample Inputs and Outputs:

```
Enter rows and column for first matrix: 3
2
Enter rows and column for second matrix: 3
2
Error! column of first matrix not equal to row of second.
Enter rows and column for first matrix: 2
3
Enter rows and column for second matrix: 3
2

Enter elements of matrix 1:
Enter elements a11: 3
Enter elements a12: -2
Enter elements a13: 5
Enter elements a21: 3
Enter elements a22: 0
Enter elements a23: 4

Enter elements of matrix 2:
Enter elements b11: 2
Enter elements b12: 3
Enter elements b21: -9
Enter elements b22: 0
Enter elements b31: 0
Enter elements b32: 4

Output Matrix:
24 29
6  25
```

Viva - Questions:

Q1. How to perform matrix addition using operator overloading concept.

Q2.Add Two Matrix Using Multi-dimensional Arrays.

Q3.Find Transpose of a Matrix.

Q4.What is Namespace?

Q5. Difference between “vector” and “array”?

Experiment 2

Aim: Write a program to perform addition of two complex numbers using constructor overloading. The first constructor which takes no argument is used to create objects which are not initialized, second which takes one argument is used to initialize real and imag parts to equal values and third which takes two argument is used to initialized real and imag to two different values.

Performance Instructions:

A class **constructor** is a special member function of a class that is executed whenever we create new objects of that class.

A constructor will have exact same name as the class and it does not have any return type at all, not even void. Constructors can be very useful for setting initial values for certain member variables.

```
class Complex
{
int real,imag;
public:
  Complex()                    // Constructor.
  {
  }
```

Parameterized Constructor

A default constructor does not have any parameter, but if you need, a constructor can have parameters. This helps you to assign initial value to an object at the time of its creation.

```
  Complex(int r)               // Parameterised constructor for equal values.
  {
  real=r;
  imag=r;
  }
  Complex(int r,int i)        // Parameterised constructor for different values.
  {
  real=r;
  imag=i;
  }
```

Copy Constructor

The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously.

```
Complex(Complex &c)       //Copy Constructor.
{
 real=c.real;
 imag=c.imag;
}
```

Addition of Complex numbers

```
Complex sum(Complex obj1,Complex obj2)

{

 Complex obj3;
 obj3.real=obj1.real+obj2.real;

 obj3.imag=obj1.imag+obj2.imag;
 return obj3;
}
```

Additional Programs:

1. Write a program to enter any number and find its factorial using constructor.

2. Write a program to generate a Fibonacci series using copy constructor.

Sample Inputs and Outputs:

PROGAM TO PERFORM ADDITION OF TWO COMPLEX NUMBERS USING CONSTRUCTOR OVERLOADING

For equal values :
Enter the equal value of real and imaginary part of number 1: 5

For different values
Enter the real and imaginary part of number 2: 10 15

The sum of two complex nos. is 15+20i

Viva - Questions:

Q1. Explain Default constructor.

Q2. When are the Global objects destroyed?

Q3. What is a virtual destructor? Explain the use of it .

Q4. Difference between a copy constructor and an assignment operator .

Q5. What are the restrictions apply to constructors and destructors?

Experiment 3

Aim: Write a program to find the greatest of two given numbers in two different classes using friend function.

Performance Instructions:

Friend Functions:
If a function is defined as a friend function then, the private and protected data of a class can be accessed using the function.

The complier knows a given function is a friend function by the use of the keyword friend.

For accessing the data, the declaration of a friend function should be made inside the body of the class (can be anywhere inside class either in private or public section) starting with keyword friend.

```
class B;        //declare class B
class A {
private:
   int a;
public:
   A(int a) {
      this->a = a;
   }
   friend int max(A a, B b);          // Friend Function  declaration
};
```

```
// Greatest of two  numbers
int max(A a, B b) {
   return (a.a > b.b ? a.a : b.b);
}
int main() {
   A a(10);
   B b(15);
   cout << "Greatest is : " << max(a, b);
   return 0;
}
```

Additional Programs:

1. Write a program to find the sum of two numbers declared in a class and display the numbers and sum using friend class.

2. Write a program to demonstrate the use of friend function with Inline assignment.

Sample Inputs and Outputs:

Enter two values: 10, 15

Greatest is: 15

Viva - Questions:

Q1. What is the difference between friend function and friend class?

Q2. Advantages of using friend classes.

Q3. Give an example for the use of volatile keyword in c++ ?.

Q4. What is a container class?.

Q5. What is function prototyping? What are its advantages?

Experiment 4

Aim: Implement a class string containing the following functions:

- Overload + operator to carry out the concatenation of strings.
- Overload = operator to carry out string copy.
- Overload <= operator to carry out the comparison of strings.
- Function to display the length of a string.
- Function tolower( ) to convert upper case letters to lower case.
- Function toupper( ) to convert lower case letters to upper case.

Performance Instructions:

Operator Overloading:

In C++, we can make operators to work for user defined classes. For example, we can overload an operator '+' in a class like String so that we can concatenate two strings by just using +.

Overloaded operators are functions with special names the keyword operator followed by the symbol for the operator being defined. Like any other function, an overloaded operator has a return type and a parameter list.

```
void operator+(char s[25])
{
    strcat(str," ");
    strcat(str,s);
}
void operator=(char s[25])
{
    strcpy(str,s);
}
void operator<=(char s[25])
{
    if(strcmp(str,s)>0)
    {
      cout<<str<<" is larger than "<<s;
    }
    else if(strcmp(str,s)<0)
    {
      cout<<s<<" is larger than "<<str;
    }
    else
```

```
    {
                cout<<str<<" is equal to "<<s;
              }

//Invoking different operators
            switch(ch)
            {
                    case '+': r+st;
                              r.display();
                              break;
                    case '=': r=st;
                              r.display();
                              break;
                    case '<': r<=st;
                              break;
                    case 'x': cout<<" Exit";
                              exit(0);
            }
```

IMPLEMENT A CLASS STRING CONTAINING THE FOLLOWING FUNCTIONS:

- ✓ FUNCTION TO DISPLAY THE LENGTH OF A STRING.
- ✓ FUNCTION TOLOWER( ) TO CONVERT UPPER CASE LETTERS TO LOWER CASE.
- ✓ FUNCTION TOUPPER( ) TO CONVERT LOWER CASE LETTERS TO UPPER CASE.

```
// Operator overloading
        void strlen()
        {
            int l=0;
            for(int i=0;str[i]!='\0';i++)
            {
             l++;
            }
            cout<<"\n The length of String is: "<<l;
        }
        void toupper()
        {
            for(int i=0;str[i]!='\0';i++)
            {
             if((str[i]>=97)&&(str[i]<=122))
             {
                str[i]=str[i]-32;
             }
            }
        }
        void tolower()          {
            for(int i=0;str[i]!='\0';i++)
            {
             if((str[i]>=65)&&(str[i]<=90))
             {
                str[i]=str[i]+32;
             }
            }
        }
```

```
// Invoking different operators
        switch(c)
        {
                case 1  : r.strlen();
                          break;
                case 2  : r.toupper();
                          r.display();
                          break;
                case 3  : r.tolower();
                          r.display();
                          break;
                case 4  : cout<<" Exit";
                          exit(0);
        }
```

Additional Programs:

1. Write a program to overload new and delete operators.

Sample Inputs and Outputs:

Enter first string: programming

Enter second string: unit

Concatenated string: programmingunit

Viva - Questions:

Q1. Difference between overloaded functions and overridden functions.

Q2. What is overloading unary operator?.

Q3. What is overloading template?

Q4. How does C++ compiler differs between overloaded postfix and prefix operators?

Q5. Which function operator cannot be over loaded

Experiment 5

Aim: Create a class called LIST with two pure virtual function store() and retrieve().To store a value call store and to retrieve call retrieve function. Derive two classes stack and queue from it and override store and retrieve.

Performance Instructions:

A virtual function is a member function which is declared within base class and is re-defined (Overriden) by derived class.When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class's version of the function.

--Virtual functions ensure that the correct function is called for an object, regardless of the type of reference (or pointer) used for function call.

--They are mainly used to achieve Runtime polymorphism

--Functions are declared with a virtual keyword in base class.

--The resolving of function call is done at Run-time.

```
// Declaration of class
class list {
public:
   virtual void store() {                    //virtual function
      cout << "\n  list class store:";
   }
   void retrieve() {
      cout << "\n  list class retrieve:";
   }
};
```

```
// Derived classes stack and queue
class stack : public list {
public:
   void store() {
      cout << "\n  stack class store:";
   }
   void retrieve() {
      cout << "\n  stack class retrieve:";
   }
};
```

Overriding

If derived class defines same function as defined in its base class, it is known as function overriding. It enables you to provide specific implementation of the function which is already provided by its base class.

```
   list obj1;
   list *p;
   cout << "\n\t P points to list:\n";
   p = &obj1;
   p->store();
   p->retrieve();
   cout << "\n\n\t P points to stack:\n";
   stack obj2;
   p = &obj2;
   p->store();
   p->retrieve();
```

Additional Programs:

1. Create a base class called SHAPE. Use this class to store two double type values. Derive two specific classes called TRIANGLE and RECTANGLE from the base class. Add to the base class, a member function getdata to initialize base class data members and another member function display to compute and display the area of figures. Make display a virtual function and redefine this function in the derived classes to suit their requirements. Using these three classes design a program that will accept driven of a TRINGLE or RECTANGLE interactively and display the area.

Sample Inputs and Outputs:
P points to list

list class store

list class retrieve

P points to stack

stack class store

stack class retrieve

Viva - Questions:

Q1. What is the purpose of the virtual keyword in C++?

Q2. What is meant by pure virtual function?

Q3. What is a virtual base class?

Q4. Explain the problem with overriding functions.

Q5. What is object slicing?

Experiment 6

Aim: Write a program to define the function template for calculating the square of given numbers with different data types.

Performance Instructions:

Function templates:

Function templates are special functions that can operate with generic types. This allows us to create a function template whose functionality can be adapted to more than one type or class without repeating the entire code for each type. In C++ this can be achieved using template parameters.

```
// Template for square function
   template <class T>
   T square(T number){
   return number * number;
   }


   // Get an integer and compute its square
   cout << "Enter an integer: ";
   int iValue;
   cin >> iValue;
   // The compiler creates int square(int) at the first
   // occurrence of a call to square with an int argument
   cout << "The square is " << square(iValue);
```

```
// Get a double and compute its square
cout << "\nEnter a double: ";
double dValue;
cin >> dValue;
// The compiler creates double square(double)at the first
// occurrence of a call to square with a double argument
cout << "The square is " << square(dValue) << endl;
return 0;
```

Additional Programs:

1. Write a program to define the function template for swapping two items of the various data types such as integer ,float,and characters.

2. Write a program to illustrate how to define and declare a class template for reading two data items from the keyboard and to find their sum.

Sample Inputs and Outputs:

Enter an integer: 3

The square is 9

Enter a double: 8.3

The square is 68.89

Viva - Questions:

Q1. Differentiate between a template class and class template.

Q2. What are the syntax and semantics for a function template?

Q3. What is the STL, standard template library?

Q4. What is meant by template parameter?

Q5. Why we use :: template-template parameter?

Experiment 7

Aim: Write a program to demonstrate the use of special functions, constructor and destructor in the class template. The program is used to find the bigger of two entered numbers.

Performance Instructions:

A class **constructor** is a special member function of a class that is executed whenever we create new objects of that class.

A constructor will have exact same name as the class and it does not have any return type at all, not even void. Constructors can be very useful for setting initial values for certain member variables.

Destructor is a member function which destructs or deletes an object.

```
class biggest
{
  private:
        int a,b;
        public:
          biggest(int a, intb);      //constructor function
                void input();
                void display();
         ~biggest()                  //destructor function
            {
                  cout<<"Objects are destroyed”<<endl
};


//Bigger of two numbers
{
        if(a>b)
        cout<<"Biggest no.:"<<a;
        else
        cout<<"Biggest no.:"<<b;
}
```

Additional Programs:

1. Write a program to illustrate how template functions can be overloaded.

Sample Inputs and Outputs:

Enter 2 nos.:33 44

Biggest no.:44

Viva - Questions:

Q1. What is the order in which constructors are called when an object of a derived class is created.

Q2. How should a constructor handle a failure?

Q3. What are shallow and deep copy?

Q4. Which constructor function is designed to copy objects of the same class type?

Q5. How many times a constructor is called in the life-time of an object?

Experiment 8

Aim: Write a program to perform the deletion of white spaces such as horizontal tab, vertical tab, space ,line feed ,new line and carriage return from a text file and store the contents of the file without the white spaces on another file.

Performance Instructions:

File: The information / data stored under a specific name on a storage device, is called a file.
Stream: It refers to a sequence of bytes.

ofstream
This data type represents the output file stream and is used to create files and to write information to files.

ifstream
This data type represents the input file stream and is used to read information from files.

fstream
This data type represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files.

```
// Deleting white spaces from a text file
ofstream outfile;
ifstream infile;
char fname1[10], fname2[10];
char ch;
cout<<"enter a file name to be copied ?\n";
cin>>fname1;
cout<<"new file name ? \n";
cin>>fname2;
infile.open(fname1);
if(infile.fail()) {
cerr<<"No such a file exists \n";
exit(1);
```

```
}
outfile.open(fname2);
if(outfile.fail()) {
cerr<<"unable to create  a file  \n";
exit(1);
}
while(!infile.eof()) {
ch = (char) infile.get();
if(ch== ' '|| ch=='t' || ch=='\n') ;
else
outfile.put(ch);
}
infile.close();
outfile.close();
return 0;
}
```

Additional Programs:

1.Write a program to read a set of lines from the keyboard and to store it on a specified file.

Sample Inputs and Outputs:

content of the input file

this is a

test program

by CSEMait

content of the output file

thisisatest programbyCSEMait

Viva - Questions:

Q1. Which stream class is used to both read and write on files ?

Q2. Which functions allow to change the location of the get and put positions ?

Q3. eof() is used to get

Q4. How to get position to the nth byte of fileObject ?

Q5. If we have object from fstream class, then what is the default mode of opening the file?

Experiment 9

Aim: Write a program to read the class object of student info such as name , age ,sex ,height and weight from the keyboard and to store them on a specified file using read() and write() functions. Again the same file is opened for reading and displaying the contents of the file on the screen.

Performance Instructions:

File: The information / data stored under a specific name on a storage device, is called a file.

Functions use in File Handling

Function	Operation
open()	To create a file
close()	To close an existing file
get()	Read a single character from a file
put()	write a single character in file.
read()	Read data from file
write()	Write data into file.

```
// array of class objects and file operations

class student_info {
protected:
char name[20] ;
int age;
char sex;
float height; float weight;
public:
void getdata();
void display();
};

// readingfromthekevboard
cout << "Enter the foilowing information \n";
for (i =O; i <= n-1; ++i){
int j = i;
cout << endl;
cout << "object= << j+l << endl;
obj[i].getdata();
```

```
// storing ontothe file
infileopen( fname,ios::out);
cout<<"storingontothefile.....\n";
for(i=O;i<=n-1;++i)
{
infile.wrIte((char*)&obj[i],sizeof(obj[i]));
}
infile.close();


/ / readingfromthefile
 infileopen(fname,ios: :in);
cout<<"readingfromthe file.....\n";
for( i= 0;i<=n-1;++i){
infile.read (char*)&obj[I ] sizeof(obj[I]));
obj[i]display ();
}
infile close();
return O;
}
```

Additional Programs:

1. Write a program to read a text file and display its contents on the screen.
2. Write a program to copy the contents of a file into another.

Sample Inputs and Outputs:

```
Enter a file name to be stored ?
Data
How many objects are to be stored?
2
Enter the following information

Object =1
Name: Shyam
Age: 27
Sex: M
Height: 172
Weight: 75

Object =2
Name: Bala
Age: 34
Sex: M
Height: 187
Weight: 82

Storing on to the file.....
Reading from the file.....
Shyam  27  M  172  75
Bala      34  M  187  82
```

Viva - Questions:

Q1. What is an inverted file.

Q2. What is seek time.

Q3. Which operator is used to insert the data into file?

Q4. How many objects are used for input and output to a string?

Q5.Which member function is used to determine whether the stream object is currently associated with a file?

Experiment 10

Aim: Write a program to raise an exception if any attempt is made to refer to an element whose index is beyond the array size.

Performance Instructions:

An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: try, catch, and throw.

1) throw − A program throws an exception when a problem shows up. This is done using a throw keyword.

2) catch − A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The catch keyword indicates the catching of an exception.

3) try − A try block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.

```
// exceptions: multiple catch blocks
 try
 {
  char * mystring;
  mystring = new char [10];
  if (mystring == NULL) throw "Allocation failure";
  for (int n=0; n<=100; n++)
  {
   if (n>9) throw n;
   mystring[n]='z';
  }
 }
 catch (int i)
 {
```

```
  cout << "Exception: ";
  cout << "index " << i << " is out of range" << endl;
}
catch (char * str)
{
  cout << "Exception: " << str << endl;
}
return 0;
}
```

Additional Programs:

1. Write a program to read two numbers and then divide first no by second no and raise an exception if second number is zero.

Sample Inputs and Outputs:

Exception: index 10 is out of range

Viva - Questions:

Q1. What will happen when a programs throws any other type of exception other than specified?

Q2. What do you mean by “No exception specification”?

Q3. Which operations don’t throw anything?

Q4. Explain terminate() and unexpected() function.

Q5. What is Asynchronous Exceptions?

Sample Viva questions

1) What is the output of printf("%d")
2)What will happen if I say delete this
3) Difference between "C structure" and "C++ structure".
4) Diffrence between a "assignment operator" and a "copy constructor"
5) What is the difference between "overloading" and "overridding"?
6) Explain the need for "Virtual Destructor".
7) Can we have "Virtual Constructors"?
8) What are the different types of polymorphism?
9) What are Virtual Functions? How to implement virtual functions in "C"
10) What are the different types of Storage classes?
11) What is Namespace?
12) What are the types of STL containers?.
13) Difference between "vector" and "array"?
14) How to write a program such that it will delete itself after execution?
15) Can we generate a C++ source code from the binary file?
16) What are inline functions?
17) What is fstream" ?
18) Explain "passing by value", "passing by pointer" and "passing by reference"
19) Have you heard of "mutable" keyword?
20) What is a "RTTI"?
21) Is there something that I can do in C and not in C++?
22) What is the difference between "calloc" and "malloc"?
23) What will happen if I allocate memory using "new" and free it using "free" or allocate sing "calloc" and free it using "delete"?
24) Difference between "printf" and "sprintf".
25) What is "map" in STL?
26) When shall I use Multiple Inheritance?
27) Explain working of printf.
28) How many lines of code you have written for a single program?
29) How to write Multithreaded applications using C++?
30) Write any small program that will compile in "C" but not in "C++"
31) What is Memory Alignment?
32) Why preincrement operator is faster than postincrement?
33) What is the difference between an ARRAY and a LIST?
34) What is faster : access the element in an ARRAY or in a LIST?
35) Define a constructor - what it is and how it might be called (2 methods).
36) Describe PRIVATE, PROTECTED and PUBLIC – the differences and give examples.
37)What is a COPY CONSTRUCTOR and when is it called .

38) Explain term POLYMORPHISM and give an example using eg. SHAPE object: If I have a base class SHAPE, how would I define DRAW methods for two objects CIRCLE and SQUARE.
39) What is the word you will use when defining a function in base class to allow this function to be a polymorphic function?
40) What are 2 ways of exporting a function from a DLL?
41) You have two pairs: new() and delete() and another pair : alloc() and free(). Explain differences between them.
42) Can we access virtual functions through objects using dot operator.
43) Can we make friend function virtual.
44) Can we overload friend function.
45) Can we inherit friend functions.
46) Friendship is not mutual in friend function.comment?
47) What are the benefits of operator overloading.
48) What does it mean that friendship is not transitive and reciprocal?
49) What is the difference between containership and inheritance?
50) What are pure virtual functions?
51) Why is it important to redefine the pure virtual function in derived class?